



# Mobile Intel® Celeron® Processor on .13 Micron Process in Micro-FCPGA Package

Specification Update

---

*April 2004*

**Notice:** The mobile Intel® Celeron® processor may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are documented in this Specification Update.

Document Number: 251309-019



INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL® PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER, AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT. Intel products are not intended for use in medical, life saving, or life sustaining applications.

Intel may make changes to specifications and product descriptions at any time, without notice.

Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined." Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them.

The Mobile Intel® Celeron® Processor may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order.

Intel, Pentium, Celeron, Xeon, NetBurst and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

\* Other names and brands may be claimed as the property of others.

Copyright © Intel Corporation 2002 - 2004



# Contents

---

Revision History .....4

Preface .....5

Errata .....11

Specification Clarifications.....27

Specification Changes .....29

Documentation Changes .....33

# Revision History

Revision Number	Description	Date
-001	Initial Release	June 2002
-002	Updated Identification Information table.	August 2002
-003	Updated Documentation Change summary by removing old items that have been incorporated into the Software Developer's Manual; Added errata V30-V32; Added Documentation Changes V3-V24.	September 2002
-004	Updated Summary of Changes; Added prefix letter W; Removed previous errata V32 as not applicable, added new errata V32; Added Documentation Changes V25-V32.	October 2002
-005	Added errata V33; Added Spec Clarification V1; Added Spec Change V1; Removed Documentation Changes that have been incorporated into the Software Developer's Manual; Added a note in Documentation Changes.	November 2002
-006	No new Errata, Documentation Change, Spec Clarification, or Spec Change. Updated note concerning the Software Developer's Manual.	December 2002
-007	Added Erratum V34. Updated summary tables to include C1 step.	January 2003
-008	Added Erratum V35.	February 2003
-009	Updated Erratum V34, added new Erratum V36	March 2003
-010	Updated codes used in summary tables. Updated errata summary table entry for V27, V28, V34, & V36. Updated tables for D1-step	April 2003
-011	Added info for C-1 step 1.6 GHz to 2.2 GHz to Table 1.	May 2003
-012	Updated Erratum V33. Added identification info for D-1 step 2.4 GHz. Updated summary table code Z.	June 2003
-013	Added Erratum V37.	July 2003
-014	Added V2 Specification Change. Added Errata V38.	August 2003
-015	Added Errata V39 and V40.	September 2003
-016	Added Errata V41.	October 2003
-017	Updated Errata V41.	November 2003
-018	Updated Errata V42.	March 2004
-019	Updated Errata V6.	April 2004

§



## Preface

---

This document is an update to the specifications contained in the following documents:

- *Mobile Intel® Celeron® Processor on .13 Micron Process in Micro-FCPGA Package Datasheet*, Reference Number 251308.

**Note:** Documentation changes for *IA-32 Intel® Architecture Software Developer's Manual* volumes 1, 2, and 3 are posted in a separate document *IA-32 Intel® Architecture Software Developer's Manual Documentation Changes*. This document has been posted to <http://developer.intel.com/>

It is intended for hardware system manufacturers and software developers of applications, operating systems, or tools and contains Errata, Documentation Changes, Specification Clarifications, and Specification Changes.

## Nomenclature

**S-Spec Number** is a five-digit code used to identify products. Products are differentiated by their unique characteristics, e.g., core speed, L2 cache size, package type, etc. as described in the processor identification information table. Care should be taken to read all notes associated with each S-Spec number.

**Errata** are design defects or errors. Errata may cause the Mobile Pentium® 4 Processor - M behavior to deviate from published specifications. Hardware and software designed to be used with any given processor must assume that all errata documented for that processor are present on all devices unless otherwise noted.

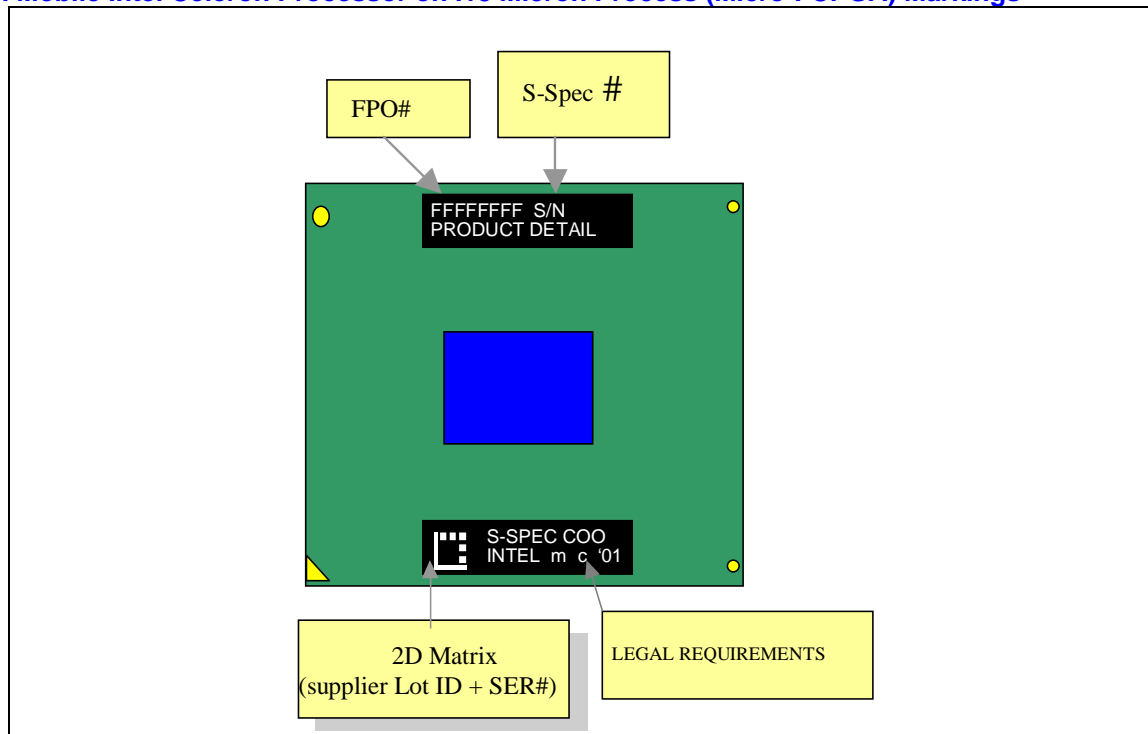
**Documentation Changes** include typos, errors, or omissions from the current published specifications. These changes will be incorporated in the next release of the specifications.

**Specification Clarifications** describe a specification in greater detail or further highlight a specification's impact to a complex design situation. These clarifications will be incorporated in the next release of the specifications.

**Specification Changes** are modifications to the current published specifications for the Pentium® 4 processor. These changes will be incorporated in the next release of the specifications.

## General Information

Figure 1. Mobile Intel Celeron Processor on .13 Micron Process (Micro-FCPGA) Markings



## Identification Information

The Mobile Intel® Celeron® processor on .13 micron process in Micro-FCPGA package can be identified by the following values:

Family <sup>1</sup>	Model <sup>2</sup>	Brand ID <sup>3</sup>
1111	0010	00001000 <sup>4</sup>
1111	0010	00001111 <sup>5</sup>

**NOTES:**

1. The Family corresponds to bits [11:8] of the EDX register after Reset, bits [11:8] of the EAX register after the CPUID instruction is executed with a 1 in the EAX register, and the generation field of the Device ID register accessible through Boundary Scan.
2. The Model corresponds to bits [7:4] of the EDX register after Reset, bits [7:4] of the EAX register after the CPUID instruction is executed with a 1 in the EAX register, and the model field of the Device ID register accessible through Boundary Scan.
3. The Brand ID corresponds to bits [7:0] of the EBX register after the CPUID instruction is executed with a 1 in the EAX register.
4. This brand id MUST be used in conjunction with CPUID = 0F24h.
5. This brand id MUST be used in conjunction with CPUID = 0F27h.



## Identification Information

S-Spec	Product Stepping	L2 Cache Size (bytes)	CPU Signature	Core Frequency	Bus Frequency	Voltage	Package	Notes
SL6FM	B0	256 K	0F24h	1.4 GHz	400 MHz	1.3 V	Micro-FCPGA	1
SL6FN	B0	256 K	0F24h	1.5 GHz	400 MHz	1.3 V	Micro-FCPGA	1
SL6M4	C1	256 K	0F27h	1.4 GHz	400 MHz	1.3 V	Micro-FCPGA	
SL6M5	C1	256 K	0F27h	1.5 GHz	400 MHz	1.3 V	Micro-FCPGA	
SL6J2	C1	256 K	0F27h	1.6 GHz	400 MHz	1.3 V	Micro-FCPGA	
SL6J3	C1	256 K	0F27h	1.7 GHz	400 MHz	1.3 V	Micro-FCPGA	
SL6J4	C1	256 K	0F27h	1.8 GHz	400 MHz	1.3 V	Micro-FCPGA	
SL6QH	C1	256 K	0F27h	2.0 GHz	400 MHz	1.3 V	Micro-FCPGA	
SL6ZW	C1	256 K	0F27h	2.2 GHz	400 MHz	1.3 V	Micro-FCPGA	
SL6VG	D1	256 K	0F29h	1.70 GHz	400 MHz	1.3 V	Micro-FCPGA	
SL6VH	D1	256 K	0F29h	1.80 GHz	400 MHz	1.3 V	Micro-FCPGA	
SL6VJ	D1	256 K	0F29h	2.00 GHz	400 MHz	1.3 V	Micro-FCPGA	
SL73Y	D1	256 K	0F29h	2.20 GHz	400 MHz	1.3 V	Micro-FCPGA	
SL75J	D1	256 K	0F29h	2.40 GHz	400 MHz	1.3 V	Micro-FCPGA	

### NOTES:

1. Based on B0-Shrink process.

## Summary of Changes

The following table indicates the Errata, Documentation Changes, Specification Clarifications, or Specification Changes that apply to Pentium 4 processors. Intel intends to fix some of the errata in a future stepping of the component, and to account for the other outstanding issues through documentation or specification changes as noted. This table uses the following notations:

### Codes Used in Summary Table

#### Stepping

X: Erratum, Specification Change or Clarification that applies to this stepping.

(No mark) or (Blank Box): This erratum is fixed in listed stepping or specification change does not apply to listed stepping.

#### Status

Doc: Document change or update that will be implemented.

PlanFix: This erratum may be fixed in a future of the product.

Fixed: This erratum has been previously fixed.

NoFix: There are no plans to fix this erratum.

PKG: This column refers to errata on the Pentium 4 processor substrate.

AP: APIC related erratum.

Shaded: This item is either new or modified from the previous version of the document.

Each Specification Update item is prefixed with a capital letter to distinguish the product. The key below details the letters that are used in Intel's microprocessor Specification Updates:

A = Intel® Pentium® II processor

B = Mobile Intel® Pentium® II processor

C = Intel® Celeron® processor

D = Intel® Pentium® II Xeon™ processor

E = Intel® Pentium® III processor

G = Intel® Pentium® III Xeon™ processor

H = Mobile Intel® Celeron® processor at 466/433/400/366/333/300 and 266 MHz

K = Mobile Intel® Pentium® III processor

M = Mobile Intel® Celeron® processor

N = Intel® Pentium® 4 processor

P = Intel® Xeon™ processor

T = Mobile Intel® Pentium® 4 Processor-M

V = Mobile Intel® Celeron® processor on .13 Micron Process in Micro-FCPGA Package

Y = Intel® Pentium® M processor

Z = Mobile Intel® Pentium® 4 processor with 533 MHz system bus

The Specification Updates for the Pentium processor, Pentium® Pro processor, and other Intel products do not use this convention

NO.	Steppings			Plans	ERRATA
	B0	C1	D1		
V1	X	X	X	NoFix	I/O restart in SMM may fail after simultaneous machine check exception (MCE)
V2	X	X	X	NoFix	MCA registers may contain invalid information if RESET# occurs and PWRGOOD is not held asserted
V3	X	X	X	NoFix	Transaction is not retried after BINIT#
V4	X	X	X	NoFix	Invalid opcode 0FFFh requires a ModRM byte
V5	X	X	X	NoFix	FSW may not be completely restored after page fault on FRSTOR or FLDENV instructions
V6	X	X	X	NoFix	The Processor Signals Page-Fault Exception (#PF) Instead of Alignment Check Exception (#AC) on an Unlocked CMPXCHG8B Instruction
V7	X	X	X	NoFix	When in no-fill mode the memory type of large pages are incorrectly forced to uncacheable
V8	X	X	X	NoFix	Processor may hang due to speculative page walks to non-existent system memory
V9	X	X	X	NoFix	The IA32_MC1_STATUS register may contain incorrect information for correctable errors





NO.	Steppings			Plans	ERRATA
	B0	C1	D1		
V10	X	X	X	NoFix	Debug mechanisms may not function as expected
V11	X	X	X	NoFix	Machine check architecture error reporting and recovery may not work as expected
V12	X	X	X	NoFix	Cascading of performance counters does not work correctly when forced overflow is enabled
V13	X	X	X	NoFix	EMON event counting of x87 loads may not work as expected
V14	X	X	X	NoFix	Speculative page fault may cause livelock
V15	X			Fixed	Incorrect data may be returned when page tables are in Write Combining (WC) memory space
V16	X	X	X	NoFix	Processor issues inconsistent transaction size attributes for locked operation
V17	X			Fixed	Multiple accesses to the same S-state L2 cache line and ECC error combination may result in loss of cache coherency
V18				Fixed	Processor may hang when resuming from Deep Sleep state
V19	X	X	X	NoFix	When the processor is in the System Management Mode (SMM), debug registers may be fully writeable
V20	X	X	X	NoFix	Associated counting logic must be configured when using Event Selection Control (ESCR) MSR
V21	X	X	X	NoFix	IA32_MC0_ADDR and IA32_MC0_MISC Registers Will Contain Invalid or Stale Data Following a Data, Address, or Response Parity Error
V22	X			Fixed	CR2 May Be Incorrect or an Incorrect Page Fault Error Code May Be Pushed Onto Stack After Execution of an LSS Instruction
V23	X			Fixed	BPM[5:3]# and GHI# V <sub>IL</sub> Do Not Meet Specification
V24	X	X	X	NoFix	Processor May Hang Under Certain Frequencies and 12.5% STPCLK# Duty Cycle
V25	X	X	X	NoFix	System May Hang if a Fatal Cache Error Causes Bus Write Line (BWL) Transaction to Occur to the Same Cache Line Address as an Outstanding Bus Read Line (BRL) or Bus Read-Invalidate Line (BRIL)
V26	X			Fixed	L2 Cache May Contain Stale Data in the Exclusive State
V27	X	X		Fixed	Re-mapping the APIC Base Address to a Value Less Than or Equal to 0xDC001000 may Cause IO and Special Cycle Failure
V28	X	X		Fixed	Erroneous BIST Result Found in EAX Register After Reset
V29	X			Fixed	Processor Does not Flag #GP on Non-zero Write to Certain MSRs
V30	X	X	X	NoFix	Simultaneous Assertion of A20M# and INIT# may Result in Incorrect Data Fetch
V31	X	X	X	PlanFix	Processor Does not Respond to Break Requests From ITP
V32	X			Fixed	Processor Signature Returns Incorrect Number of ITLB Entries
V33	X	X	X	NoFix	A Write to APIC Registers Sometimes May Appear to Have Not Occurred
V34		X		Fixed	Store to Load Data Forwarding may Result in Switched Data Bytes
V35	X	X	X	NoFix	Parity Error in the L1 Cache may Cause the Processor to Hang

NO.	Steppings			Plans	ERRATA
	B0	C1	D1		
V36	X	X		Fixed	The TCK Input in the Test Access Port (TAP) is Sensitive to Low Clock Edge Rates and Prone to Noise Coupling Onto TCK's Rising or Falling Edges
V37	X	X	X	NoFix	The State of the Resume Flag (RF Flag) in a Task-State Segment (TSS) May be Incorrect
V38	X	X	X	NoFix	Changes to CR3 Register do not Fence Pending Instruction Page Walks
V39	X	X	X	NoFix	System Bus Interrupt Messages without Data that Receive a HardFailure Response May Hang the Processor
V40	X	X	X	NoFix	Memory Type of the Load Lock Different from its Corresponding Store Unlock
V41	X	X	X	NoFix	A 16-bit Address Wrap Resulting from a Near Branch (Jump or Call) May Cause an Incorrect Address to Be Reported to the #GP Exception Handler
V42	X	X	X	NoFix	Incorrect Debug Exception (#DB) May Occur When a Data Breakpoint is set on an FP Instruction

NUMBER	SPECIFICATION CHANGE
V1	Context ID feature added to CPUID Feature/IA32_MISC_Enable Registers
V2	BR0# Maximum Hold Time Specification Change

NUMBER	SPECIFICATION CLARIFICATIONS
V1	Clarifying DBI# Definition for all Processors with Intel NetBurst® Microarchitecture

NUMBER	DOCUMENTATION CHANGES
	There are no Documentation Changes in this Specification Update revision.

§

## Errata

---

### V1. I/O Restart in SMM may Fail after Simultaneous Machine Check Exception (MCE)

**Problem:** If an I/O instruction (IN, INS, REP INS, OUT, OUTS, or REP OUTS) is being executed, and if the data for this instruction becomes corrupted, the processor will signal a Machine Check Exception (MCE). If the instruction is directed at a device that is powered down, the processor may also receive an assertion of SMI#. Since MCEs have higher priority, the processor will call the MCE handler, and the SMI# assertion will remain pending. However, upon attempting to execute the first instruction of the MCE handler, the SMI# will be recognized and the processor will attempt to execute the SMM handler. If the SMM handler is completed successfully, it will attempt to restart the I/O instruction, but will not have the correct machine state, due to the call to the MCE handler.

**Implication:** A simultaneous MCE and SMI# assertion may occur for one of the I/O instructions above. The SMM handler may attempt to restart such an I/O instruction, but will have an incorrect state due to the MCE handler call, leading to failure of the restart and shutdown of the processor.

**Workaround:** If a system implementation must support both SMM and board I/O restart, the first thing the SMM handler code should do is check for a pending MCE. If there is an MCE pending, the SMM handler should immediately exit via an RSM instruction and allow the MCE handler to execute. If there is no MCE pending, the SMM handler may proceed with its normal operation.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

### V2. MCA Registers may Contain Invalid Information if RESET# Occurs and PWRGOOD is not Held Asserted

**Problem:** This erratum can occur as a result either of the following events:

- PWRGOOD is de-asserted during a RESET# assertion causing internal glitches that may result in the possibility that the MCA registers latch invalid information.
- Or during a reset sequence if the processor's power remains valid regardless of the state of PWRGOOD, and RESET# is re-asserted before the processor has cleared the MCA registers, the processor will begin the reset process again but may not clear these registers.

**Implication:** When this erratum occurs, the information in the MCA registers may not be reliable.

**Workaround:** Ensure that PWRGOOD remains asserted throughout any RESET# assertion and that RESET# is not re-asserted while PWRGOOD is de-asserted.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

### **V3. Transaction is not Retried after BINIT#**

**Problem:** If the first transaction of a locked sequence receives a HITM# and DEFER# during the snoop phase it should be retried and the locked sequence restarted. However, if BINIT# is also asserted during this transaction, it will not be retried.

**Implication:** When this erratum occurs, locked transactions will unexpectedly not be retried.

**Workaround:** None identified

**Status:** For the steppings affected, see the *Summary Table of Changes*.

### **V4. Invalid Opcode 0FFFh Requires a ModRM Byte**

**Problem:** Some invalid opcodes require a ModRM byte (or other following bytes), while others do not. The invalid opcode 0FFFh did not require a ModRM byte in previous generation Intel architecture processors, but does in the Pentium 4 processor.

**Implication:** The use of an invalid opcode 0FFFh without the ModRM byte may result in a page or limit fault on the Pentium 4 processor.

**Workaround:** Use a ModRM byte with invalid 0FFFh opcode.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

### **V5. FSW may not be Completely Restored after Page Fault on FRSTOR or FLDENV Instructions**

**Problem:** If the FPU operating environment or FPU state (operating environment and register stack) being loaded by an FLDENV or FRSTOR instruction wraps around a 64Kbyte or 4Gbyte boundary and a page fault (#PF) or segment limit fault (#GP or #SS) occurs on the instruction near the wrap boundary, the upper byte of the FPU status word (FSW) might not be restored. If the fault handler does not restart program execution at the faulting instruction, stale data may exist in the FSW.

**Implication:** When this erratum occurs, stale data will exist in the FSW.

**Workaround:** Ensure that the FPU operating environment and FPU state do not cross 64Kbyte or 4Gbyte boundaries. Alternately, ensure that the page fault handler restarts program execution at the faulting instruction after correcting the paging problem.

**Status:** For the steppings affected, see the *Summary Table of Changes*.



<b>V6.</b>	<b>The Processor Signals Page-Fault Exception (#PF) Instead of Alignment Check Exception (#AC) on an Unlocked CMPXCHG8B Instruction</b>
<b>Problem:</b>	If a Page-Fault Exception (#PF) and Alignment Check Exception (#AC) both occur for an unlocked CMPXCHG8B instruction, then #PF will be flagged.
<b>Implication:</b>	Software that depends on the Alignment Check Exception (#AC) before the Page-Fault Exception (#PF) will be affected since #PF is signaled in this case.
<b>Workaround:</b>	Remove the software's dependency on #AC having precedence over #PF. Alternately, correct the page fault in the page fault handler and then restart the faulting instruction.
<b>Status:</b>	For the steppings affected, see the Summary Table of Changes
<b>V7.</b>	<b>When in No-Fill Mode the Memory Type of Large Pages are Incorrectly Forced to Uncacheable</b>
<b>Problem:</b>	When the processor is operating in No-Fill Mode (CR0.CD=1), the paging hardware incorrectly forces the memory type of large (PSE-4M and PAE-2M) pages to uncacheable (UC) memory type regardless of the MTRR settings. By forcing the memory type of these pages to UC, load operations, which should hit valid data in the L1 cache, are forced to load the data from system memory. Some applications will lose the performance advantage associated with the caching permitted by other memory types.
<b>Implication:</b>	This erratum may result in some performance degradation when using no-fill mode with large pages.
<b>Workaround:</b>	None identified
<b>Status:</b>	For the steppings affected, see the <i>Summary Table of Changes</i> .
<b>V8.</b>	<b>Processor may Hang due to Speculative Page Walks to Non-Existent System Memory</b>
<b>Problem:</b>	A load operation that misses the Data Translation Lookaside Buffer (DTLB) will result in a page-walk. If the page-walk loads the Page Directory Entry (PDE) from cacheable memory and that PDE load returns data that points to a valid Page Table Entry (PTE) in uncacheable memory the processor will access the address referenced by the PTE. If the address referenced does not exist the processor will hang with no response from system memory.
<b>Implication:</b>	Processor may hang due to speculative page walks to non-existent system memory.
<b>Workaround:</b>	Page directories and page tables in UC memory space that are marked valid must point to physical addresses that will return a data response to the processor.
<b>Status:</b>	For the steppings affected, see the <i>Summary Table of Changes</i> .

## V9. The IA32\_MC1\_STATUS Register may Contain Incorrect Information for Correctable Errors

**Problem:** When a speculative load operation hits the L2 cache and receives a correctable error, the IA32\_MC1\_STATUS register may be updated with incorrect information. The IA32\_MC1\_STATUS register should not be updated for speculative loads.

**Implication:** When this erratum occurs, the IA32\_MC1\_STATUS register will contain incorrect information for correctable errors.

**Workaround:** None identified

**Status:** For the steppings affected, see the *Summary Table of Changes*.

## V10. Debug Mechanisms may not Function as Expected

**Problem:** Certain debug mechanisms may not function as expected on the processor. The cases are as follows:

- When the following conditions occur: 1) An FLD instruction signals a stack overflow or underflow, 2) the FLD instruction splits a page-boundary or a 64 byte cache line boundary, 3) the instruction matches a Debug Register on the high page or cache line respectively, and 4) the FLD has a stack fault and a memory fault on a split access, the processor will only signal the stack fault and the debug exception will not be taken.
- When a data breakpoint is set on the ninth and/or tenth byte(s) of a floating point store using the Extended Real data type, and an unmasked floating point exception occurs on the store, the breakpoint will not be captured.
- When any instruction has multiple debug register matches, and any one of those debug registers is enabled in DR7, all of the matches should be reported in DR6 when the processor goes to the debug handler. This is not true during a REP instruction. As an example, during execution of a REP MOVSW instruction the first iteration a load matches DR0 and DR2 and sets DR6 as FFFF0FF5h. On a subsequent iteration of the instruction, a load matches only DR0. The DR6 register is expected to still contain FFFF0FF5h, but the processor will update DR6 to FFFF0FF1h.
- A Data breakpoint that is set on a load to uncacheable memory may be ignored due to an internal segment register access conflict. In this case the system will continue to execute instructions, bypassing the intended breakpoint. Avoiding having instructions that access segment descriptor registers, e.g., LGDT, LIDT close to the UC load, and avoiding serialized instructions before the UC load will reduce the occurrence of this erratum.

**Implication:** Certain debug mechanisms do not function as expected on the processor.

**Workaround:** None identified

**Status:** For the steppings affected, see the *Summary Table of Changes*.



## V11. Machine Check Architecture Error Reporting and Recovery may not Work as Expected

**Problem:** When the processor detects errors it should attempt to report and/or recover from the error. In the situations described below, the processor does not report and/or recover from the error(s) as intended.

- When a transaction is deferred during the snoop phase and subsequently receives a Hard Failure response, the transaction should be removed from the bus queue so that the processor may proceed. Instead, the transaction is not properly removed from the bus queue, the bus queue is blocked, and the processor will hang.
- When a hardware prefetch results in an uncorrectable tag error in the L2 cache, IA32\_MC0\_STATUS.UNCOR and IA32\_MC0\_STATUS.PCC are set but no Machine Check Exception (MCE) is signaled. No data loss or corruption occurs because the data being prefetched has not been used. If the data location with the uncorrectable tag error is subsequently accessed, an MCE will occur. However, upon this MCE, or any other subsequent MCE, the information for that error will not be logged because IA32\_MC0\_STATUS.UNCOR has already been set and the MCA status registers will not contain information about the error which caused the MCE assertion but instead will contain information about the prefetch error event.
- When the reporting of errors is disabled for Machine Check Architecture (MCA) Bank 2 by setting all IA32\_MC2\_CTL register bits to 0, uncorrectable errors should be logged in the IA32\_MC2\_STATUS register but no machine-check exception should be generated. Uncorrectable loads on bank 2, which would normally be logged in the IA32\_MC2\_STATUS register, are not logged.
- When one half of a 64 byte instruction fetch from the L2 cache has an uncorrectable error and the other 32 byte half of the same fetch from the L2 cache has a correctable error, the processor will attempt to correct the correctable error but cannot proceed due to the uncorrectable error. When this occurs the processor will hang.
- When an L1 cache parity error occurs, the cache controller logic should write the physical address of the data memory location that produced that error into the IA32\_MC1\_ADDR register. In some instances of a parity error on a load operation that hits the L1 cache, however, the cache controller logic may write the physical address from a subsequent load or store operation into the IA32\_MC1\_ADDR register.
- The local xAPIC has an Error Status Register which records all errors which it detects. Bit 6 of this register, the Receive Illegal Vector bit, is set when the local xAPIC detects an illegal vector in a message that it received. When an illegal vector error is received on the same internal clock that the error status register is being written due to a previous error, bit 6 does not get set and illegal vector errors are not flagged.
- If an instruction fetch results in an uncorrectable error and there is also a debug breakpoint at this address, the processor will livelock and the uncorrectable error will not be logged in the machine check registers.
- The MCA Overflow bit should be set when an uncorrectable error resides within the register bank (valid bit is already set) and any subsequent errors occur. The Overflow bit being set indicates that more than one error has occurred. Because of this erratum, if any further errors occur, the MCA Overflow bit will not be updated; thereby incorrectly indicating only one error has been received.

**Implication:** The processor is unable to correctly report and/or recover from certain errors.

**Workaround:** None identified

**Status:** For the steppings affected, see the *Summary Table of Changes*.

## **V12. Cascading of Performance Counters does not work Correctly when Forced Overflow is Enabled**

**Problem:** The performance counters are organized into pairs. When the CASCADE bit of the Counter Configuration Control Register (CCCR) is set, a counter that overflows will continue to count in the other counter of the pair. The FORCE\_OVF bit forces the counters to overflow on every non-zero increment. When the FORCE\_OVF bit is set, the counter overflow bit will be set but the counter no longer cascades.

**Implication:** The performance counters do not cascade when the FORCE\_OVF bit is set.

**Workaround:** None identified

**Status:** For the steppings affected, see the *Summary Table of Changes*.

## **V13. EMON Event Counting of x87 Loads may not Work as Expected**

**Problem:** If a performance counter is set to count x87 loads and floating point exceptions are unmasked, the FPU Operand Data Pointer (FDP) may become corrupted.

**Implication:** When this erratum occurs, the FPU Operand Data Pointer (FDP) may become corrupted.

**Workaround:** This erratum will not occur with floating point exceptions masked. If floating point exceptions are unmasked, then performance counting of x87 loads should be disabled.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

## **V14. Speculative Page Fault may cause Livelock**

**Problem:** If the processor detects a page fault which is corrected before the operating system page fault handler can be called e.g. DMA activity modifies the page tables and the corrected page tables are left in a non-accessed or not dirty state, the processor may livelock. Intel has not been able to reproduce this erratum with commercial software.

**Implication:** Should this erratum be encountered the processor will livelock resulting in a system hang or operating system failure.

**Workaround:** It is possible for the BIOS to contain a workaround for this erratum.

**Status:** For the steppings affected, see the *Summary Table of Changes*.





**V15. Incorrect Data May be Returned When Page Tables Are In Write Combining Memory (WC) Space**

**Problem:** If page directories and/or page tables are located in Write Combining (WC) memory, speculative loads to cacheable memory may complete with incorrect data.

**Implication:** Cacheable loads to memory mapped using page tables located in write combining memory may return incorrect data. Intel has not been able to reproduce this erratum with commercially available software.

**Workaround:** Do not place page directories and/or page tables in WC memory.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

**V16. Processor Issues Inconsistent Transaction Size Attributes for Locked Operation**

**Problem:** When the processor is in the Page Address Extension (PAE) mode and detects the need to set the Access and/or Dirty bits in the page directory or page table entries, the processor sends an 8 byte load lock onto the system bus. A subsequent 8 byte store unlock is expected, but instead a 4 byte store unlock occurs. Correct data is provided since only the lower bytes change, however external logic monitoring the data transfer may be expecting an 8 byte store unlock.

**Implication:** No known commercially available chipsets are affected by this erratum.

**Workaround:** None identified.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

**V17. Multiple Accesses to the Same S-State L2 Cache Line and ECC Error Combination May Result in Loss of Cache Coherency**

**Problem:** When a Read for Ownership (RFO) cycle has a 64 bit address match with an outstanding read hit on a line in the L2 cache which is in the S-state AND that line contains an ECC error, the processor should recycle the RFO until the ECC error is handled. Due to this erratum, the processor does not recycle the RFO and attempts to service both the RFO and the read hit at the same time.

**Implication:** When this erratum occurs, cache may become incoherent.

**Workaround:** None identified.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

**V18. Processor May Hang When Resuming From Deep Sleep State**

**Problem:** When resuming from the Deep Sleep state the address strobe signals (ADSTB[1:0]#) may become out of phase with respect to the system bus clock (BCLK).

**Implication:** When this erratum occurs, the processor will hang.

**Workaround:** The system BIOS should prevent the processor from going to the Deep Sleep state.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

## **V19. When the Processor is in the System Management Mode (SMM), Debug Registers May be Fully Writeable**

**Problem:** When in System Management Mode (SMM), the processor executes code and stores data in the SMRAM space. When the processor is in this mode and writes are made to DR6 and DR7, the processor should block writes to the reserved bit locations. Due to this erratum, the processor may not block these writes. This may result in invalid data in the reserved bit locations.

**Implication:** Reserved bit locations within DR6 and DR7 may become invalid.

**Workaround:** Software may perform a read/modify/write when writing to DR6 and DR7 to ensure that the values in the reserved bits are maintained.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

## **V20. Associated Counting Logic Must be Configured When Using Event Selection Control (ESCR) MSR**

**Problem:** ESCR MSRs allow software to select specific events to be counted, with each ESCR usually associated with a pair of performance counters. ESCRs may also be used to qualify the detection of at-retirement events that support precise-event-based sampling (PEBS). A number of performance metrics that support PEBS require a 2nd ESCR to tag uops for the qualification of at-retirement events. (The first ESCR is required to program the at-retirement event.) Counting is enabled via counter configuration control registers (CCCR) while the event count is read from one of the associated counters. When counting logic is configured for the subset of at-retirement events that require a 2nd ESCR to tag uops, at least one of the CCCRs in the same group of the 2nd ESCR must be enabled.

**Implication:** If no CCCR/counter is enabled in a given group, the ESCR in that group that is programmed for tagging uops will have no effect. Hence a subset of performance metrics that require a 2nd ESCR for tagging uops may result in 0 count.

**Workaround:** Ensure that at least one CCCR/counter in the same group as the tagging ESCR is enabled for those performance metrics that require two ESCRs and tagging uops for at-retirement counting.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

## **V21. IA32\_MC0\_ADDR and IA32\_MC0\_MISC Registers Will Contain Invalid or Stale Data Following a Data, Address, or Response Parity Error**

**Problem:** If the processor experiences a data, address, or response parity error, the ADDR\_V and MISC\_V bits of the IA32\_MC0\_STATUS register are set, but the IA32\_MC0\_ADDR and IA32\_MC0\_MISC registers are not loaded with data regarding the error.

**Implication:** When this erratum occurs, the IA32\_MC0\_ADDR and IA32\_MC0\_MISC registers will contain invalid or stale data.

**Workaround:** Ignore any information in the IA32\_MC0\_ADDR and IA32\_MC0\_MISC registers after a data, address or response parity error.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

## **V22. CR2 May Be Incorrect or an Incorrect Page Fault Error Code May Be Pushed onto Stack After Execution of an LSS Instruction**

**Problem:** Under certain timing conditions, the internal load of the selector portion of the LSS instruction may complete with potentially incorrect speculative data before the load of the offset portion of the address completes. The incorrect data is corrected before the completion of the LSS instruction but the value of CR2 and the error code pushed on the stack are reflective of the speculative state. Intel has not observed this erratum with commercially available software.

**Implication:** When this erratum occurs, the contents of CR2 may be off by two, or an incorrect page fault error code may be pushed onto the stack.

**Workaround:** It is possible for the BIOS to contain a workaround for this erratum.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

## **V23. BPM[5:3]# and GHI# VIL Does Not Meet Specification**

**Problem:** The  $V_{IL}$  for BPM[5:3]# and GHI# is specified as  $0.9 * GTLREF$  [V]. Due to this erratum the  $V_{IL}$  for these signals is  $0.9 * GTLREF - .075$  [V].

**Implication:** The processor requires a lower input voltage than specified to recognize a low voltage on the BPM[5:3]# and GHI# signals.

**Workaround:** When intending to drive the BPM[5:3]# or GHI# signals low, ensure that the system provides a voltage lower than  $0.9 * GTLREF - .075$  [V].

**Status:** For the steppings affected, see the *Summary Table of Changes*.

## **V24. Processor May Hang Under Certain Frequencies and 12.5% STPCLK# Duty Cycle**

**Problem:** If a system de-asserts STPCLK# at a 12.5% duty cycle, the processor is running below 2 GHz, and the processor thermal control circuit (TCC) on-demand clock modulation is active, the processor may hang. This erratum does not occur under the automatic mode of the TCC.

**Implication:** When this erratum occurs, the processor will hang.

**Workaround:** If use of the on-demand mode of the processor's TCC is desired in conjunction with STPCLK# modulation, then assure that STPCLK# is not asserted at a 12.5% duty cycle.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

## **V25. System May Hang if a Fatal Cache Error Causes Bus Write Line (BWL) Transaction to Occur to the Same Cache Line Address as an Outstanding Bus Read Line (BRL) or Bus Read-Invalidate Line (BRIL)**

**Problem:** A processor internal cache fatal data ECC error may cause the processor to issue a BWL transaction to the same cache line address as an outstanding BRL or BRIL. As it is not typical behavior for a single

processor to have a BWL and a BRL/BRIL concurrently outstanding to the same address, this may represent an unexpected scenario to system logic within the chipset.

**Implication:** The processor may not be able to fully execute the machine check handler in response to the fatal cache error if system logic does not ensure forward progress on the system bus under this scenario.

**Workaround:** System logic should ensure completion of the outstanding transactions. Note that during recovery from a fatal data ECC error, memory image coherency of the BWL with respect to BRL/BRIL transactions is not important. Forward progress is the primary requirement.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

## V26. L2 Cache May Contain Stale Data in the Exclusive State

**Problem:** If a cacheline (A) is in Modified (M) state in the write-combining (WC) buffers and in the Invalid (I) state in the L2 cache and its adjacent sector (B) is in the Invalid (I) state and the following scenario occurs:

1. A read to B misses in the L2 cache and allocates cacheline B and its associated second-sector prefetch into an almost full bus queue,
2. A Bus Read Line (BRL) to cacheline B completes with HIT# and fills data in Shared (S) state,

The bus queue full condition causes the prefetch to cacheline A to be cancelled, cacheline A will remain M in the WC buffers and I in the L2 while cacheline B will be in the S state.

Then, if the further conditions occur:

1. Cacheline A is evicted from the WC Buffers to the bus queue which is still almost full,
2. A hardware prefetch Read for Ownership (RFO) to cacheline B, hits the S state in the L2 and observes cacheline A in the I state, allocates both cachelines,
3. An RFO to cacheline A completes before the WC Buffers write modified data back, filling the L2 with stale data,
4. The writeback from the WC Buffers completes leaving stale data, for cacheline A, in the Exclusive (E) state in the L2 cache.

**Implication:** Stale data may be consumed leading to unpredictable program execution. Intel has not been able to reproduce this erratum with commercial software.

**Workaround:** It is possible for BIOS to contain a workaround for this erratum.

**Status:** For the steppings affected, see the *Summary Table of Changes*.



**V27. Re-mapping the APIC Base Address to a Value Less Than or Equal to 0xDC001000 may Cause IO and Special Cycle Failure**

**Problem:** Remapping the APIC base address from its default can cause conflicts with either I/O or special cycle bus transactions.

**Implication:** Either I/O or special cycle bus transactions can be redirected to the APIC, instead of appearing on the front-side bus.

**Workaround:** Use any APIC base addresses above 0xDC001000 as the relocation address.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

**V28. Erroneous BIST Result Found in EAX Register After Reset**

**Problem:** The processor may show an erroneous BIST (built-in self test) result in the EAX register bit 0 after reset.

**Implication:** When this erratum occurs, an erroneous BIST failure will be reported in the EAX register bit 0, however this failure can be ignored since it is not accurate.

**Workaround:** It is possible for BIOS to workaround this issue by masking off bit 0 in the EAX register where BIST results are written.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

**V29. Processor Does not Flag #GP on Non-zero Write to Certain MSRs**

**Problem:** When a non-zero write occurs to the upper 32 bits of IA32\_CR\_SYSENTER\_EIP or IA32\_CR\_SYSENTER\_ESP, the processor should indicate a general protection fault by flagging #GP. Due to this erratum, the processor does not flag #GP.

**Implication:** The processor unexpectedly does not flag #GP on a non-zero write to the upper 32 bits of IA32\_CR\_SYSENTER\_EIP or IA32\_CR\_SYSENTER\_ESP. No known commercially available operating system has been identified to be affected by this erratum.

**Workaround:** None identified.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

**V30. Simultaneous Assertion of A20M# and INIT# may Result in Incorrect Data Fetch**

**Problem:** If A20M# and INIT# are simultaneously asserted by software, followed by a data access to the 0xFFFFFFF memory region, with A20M# still asserted, incorrect data will be accessed. With A20M# asserted, an access to 0xFFFFFFF should result in a load from physical address 0xFFEFFFF. However, in the case of A20M# and INIT# being asserted together, the data load will actually be from the physical address 0xFFFFFFF. Code accesses are not effected by this erratum.

**Implication:** Processor may fetch incorrect data, resulting in BIOS failure.

**Workaround:** Deasserting and reasserting A20M# prior to the data access will workaround this erratum.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

### **V31. Processor Does not Respond to Break Requests From ITP**

**Problem:** On power-up and low-power state transitions, the processor's TAP circuitry may remain in the Tap-Logic-Reset(TLR) state.

**Implication:** The ITP is unable to cause a break on reset in the processor, which may prevent the loading of processor and chipset registers, or affect the ability to debug from cold boot and low power transitions.

**Workaround:** None identified.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

### **V32. CPUID Instruction Returns Incorrect Number of ITLB Entries**

**Problem:** When CPUID is executed with EAX = 2 it should return a value of 51h in EAX[15:8] to indicate that the Instruction Translation Lookaside Buffer (ITLB) has 128 entries. Due to this erratum, the processor returns 50h (64 entries).

**Implication:** Software may incorrectly report the number of ITLB entries. Operation of the processor is not affected.

**Workaround:** None identified.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

### **V33. A Write to APIC Registers Sometimes May Appear to Have Not Occurred**

**Problem:** In respect to the retirement of instructions, stores to the uncacheable memory-based APIC register space are handled in a non-synchronized way. For example if an instruction that masks the interrupt flag, e.g. CLI, is executed soon after an uncacheable write to the Task Priority Register (TPR) that lowers the APIC priority, the interrupt masking operation may take effect before the actual priority has been lowered. This may cause interrupts whose priority is lower than the initial TPR, but higher than the final TPR, to not be serviced until the interrupt flag is finally cleared, i.e. by STI instruction. Interrupts will remain pending and are not lost.

**Implication:** In this example the processor may allow interrupts to be accepted but may delay their service.

**Workaround:** This non-synchronization can be avoided by issuing an APIC register read after the APIC register write. This will force the store to the APIC register before any subsequent instructions are executed. No commercial operating system is known to be impacted by this erratum.

**Status:** For the steppings affected, see the *Summary Table of Changes*.



#### **V34. Store to Load Data Forwarding may Result in Switched Data Bytes**

**Problem:** If in a short window after an instruction that updates a segment register has executed but not yet retired, there is a load occurring to an address matches a recent previous store operation but the data size is smaller than the store, the resulting data forwarded from the store to the load may have some of the lower bytes switched.

**Implication:** If this erratum occurs, the processor may execute with incorrect data.

**Workaround:** It is possible for the BIOS to contain a workaround for this erratum.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

#### **V35. Parity Error in the L1 Cache may Cause the Processor to Hang**

**Problem:** If a locked operation accesses a line in the L1 cache that has a parity error, it is possible that the processor may hang while trying to evict the line.

**Implication:** If this erratum occurs, it may result in a system hang. Intel has not observed this erratum with any commercially available software.

**Workaround:** None.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

#### **V36. The TCK Input in the Test Access Port (TAP) is Sensitive to Low Clock Edge Rates and Prone to Noise Coupling Onto TCK's Rising or Falling Edges**

**Problem:** TCK is susceptible to double clocking when low-amplitude noise is riding on TCK edge, while it is crossing the receiver's transition region. TAP failures tend to increase with increases in background system noise.

**Implication:** This only impacts JTAG/TAP accesses to the processor. Other bus accesses are not affected.

**Workaround:** To minimize the effects of this issue, reduce noise on the TCK-net at the processor relative to ground, and position TCK relative to BCLK to minimize the TAP error rate. Decreasing rise times to under 800ps reduced the failure rate but does not stop all failures.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

### **V37. The State of the Resume Flag (RF Flag) in a Task-State Segment (TSS) May be Incorrect**

**Problem:** After executing a JMP instruction to the next (or other) task through a hardware task switch, it is possible for the state of the RF flag (in the EFLAGS register image) to be incorrect.

**Implication:** The RF flag is normally used for code breakpoint management during debug of an application. It is not typically used during normal program execution. Code breakpoints or single step debug behavior in the presence of hardware task switches, therefore, may be unpredictable as a result of this erratum. This erratum has not been observed in commercially available software.

**Workaround:** None.

**Status:** For the steppings affected, see the Summary Table of Changes.

### **V38. Changes to CR3 Register do not Fence Pending Instruction Page Walks**

**Problem:** When software writes to the CR3 register, it is expected that all previous/outstanding code, data accesses and page walks are completed using the previous value in CR3 register. Due to this erratum, it is possible that a pending instruction page walk is still in progress, resulting in an access (to the PDE portion of the page table) that may be directed to an incorrect memory address.

**Implication:** The results of the access to the PDE will not be consumed by the processor so the return of incorrect data is benign. However, the system may hang if the access to the PDE does not complete with data (e.g. infinite number of retries).

**Workaround:** It is possible for the BIOS to have a workaround for this erratum.

**Status:** For the steppings affected, see the *Summary Table of Changes*.

### **V39. System Bus Interrupt Messages without Data that Receive a HardFailure Response May Hang the Processor**

**Problem:** When a system bus agent (processor or chipset) issues an interrupt transaction without data onto the system bus and the transaction receives a HardFailure response, a potential processor hang can occur. The processor, which generates an inter-processor interrupt (IPI) that receives the HardFailure response, will still log the MCA error event cause as HardFailure, even if the APIC causes a hang. Other processors, which are true targets of the IPI, will also hang on hardfail-without-data, but will not record an MCA HardFailure event as the cause. If a HardFailure response occurs on a system bus interrupt message with data, the APIC will complete the operation so as not to hang the processor.

**Implication:** The processor may hang.

**Workaround:** None identified.

**Status:** For the steppings affected, see the *Summary Tables of Changes*.



#### **V40. Memory Type of the Load Lock Different from its Corresponding Store Unlock**

**Problem:** A use-once protocol is employed to ensure that the processor in a multi-agent system may access data that is loaded into its cache on a Read-for-Ownership operation at least once before it is snooped out by another agent. This protocol is necessary to avoid a multi-agent livelock scenario in which the processor cannot gain ownership of a line and modify it before that data is snooped out by another agent. In the case of this erratum, split load lock instructions incorrectly trigger the use-once protocol. A load lock operation accesses data that splits across a page boundary with both pages of WB memory type. The use-once protocol activates and the memory type for the split halves get forced to UC. Since use-once does not apply to stores, the store unlock instructions go out as WB memory type. The full sequence on the bus is: locked partial read (UC), partial read (UC), partial write (WB), locked partial write (WB). The use-once protocol should not be applied to load locks.

**Implication:** When this erratum occurs, the memory type of the load lock will be different than the memory type of the store unlock operation. This behavior (load locks and store unlocks having different memory types) does not introduce any functional failures such as system hangs or memory corruption.

**Workaround:** None identified.

**Status:** For the steppings affected, see the *Summary Tables of Changes*.

#### **V41. A 16-bit Address Wrap Resulting from a Near Branch (Jump or Call) May Cause an Incorrect Address to Be Reported to the #GP Exception Handler**

**Problem:** If a 16-bit application executes a branch instruction that causes an address wrap to a target address outside of the code segment, the address of the branch instruction should be provided to the general protection exception handler. It is possible that, as a result of this erratum, that the general protection handler may be called with the address of the branch target.

**Implication:** The 16-bit software environment which is affected by this erratum, will see that the address reported by the exception handler points to the target of the branch, rather than the address of the branch instruction.

**Workaround:** None identified

**Status:** For the steppings affected, see the *Summary Tables of Changes*.

#### **V42. FOPCODE Routine signals Debug Exception (#DB) if a Data Breakpoint is set on an FP Instruction**

**Problem:** The default Microcode Floating Point Event Handler routine executes a series of loads to obtain data about the FP instruction that is causing the FP event. If a data breakpoint is set on the instruction causing the FP event, the load in the microcode routine will trigger the data breakpoint resulting in a Debug Exception.

**Implication:** An incorrect Debug Exception (#DB) may occur if data breakpoint is placed on an FP instruction. Intel has not observed this erratum with any commercially available software or system.

**Workaround:** None identified.

**Status:** For the steppings affected, see the *Summary Table of Changes*

§





## Specification Clarifications

---

The Specification Clarifications listed in this section apply to the following documents:

- *Mobile Intel® Celeron® Processor on .13 Micron Process in Micro-FCPGA Package Datasheet* (Order Number 251308)

All Specification Clarifications will be incorporated into a future version of the appropriate mobile Intel Celeron processor on 0.13 micron process in Micro-FCPGA Package documentation.

### **V1. Clarifying DBI# Definition for all Processors with Intel NetBurst® Micro-architecture**

The definition of DBI# signals will be clarified for Pentium 4 processors with Intel NetBurst micro-architecture. The new definition will state: DBI[3:0]# are source synchronous and indicate the polarity of the D[63:0]# signals. The DBI[3:0]# signals are activated when the data on the data bus is inverted. If more than half of the data bits, within a 16-bit group, would have been asserted electrically low, the bus agent may invert the data bus signals for that particular sub-phase for that 16-bit group.

§





## Specification Changes

The Specification Changes listed in this section apply to the following documents:

- *Mobile Intel® Celeron® Processor on .13 Micron Process in Micro-FCPGA Package Datasheet* (Order Number 251308)

All Specification Changes will be incorporated into a future version of the appropriate mobile Intel Celeron processor on 0.13 micron process in Micro-FCPGA package documentation.

### V1. Context ID Feature Added to the CPUID Instruction Feature Flags/IA32\_MISC\_ENABLE Registers

IA32\_MISC\_ENABLE register, bit 24 status has changed from Reserved to the following definition:

#### IA32\_MISC\_ENABLE – Miscellaneous Enables Register, bit # 24

MSR Address: 01A0h Accessed as a Qword  
Default Value: High Dword XXXX XXXXh  
Low Dword XXXX XXXX XXXX XXXX XXXX XX00 X0X0 0001b  
Access: Read/Write  
Type: Shared

IA32\_MISC\_ENABLE is a 64-bit register accessed only when referenced as a Qword through a RDMSR or WRMSR instruction.

Bit 24 of the IA32\_MISC\_ENABLE status has changed from Reserved to the following:

Bit	Descriptions
24	<b>L1 Data Cache Context Mode (R/W).</b> When set to a '1' this bit places the L1 Data Cache into shared mode. When set to a '0' (default) this bit places the L1 Data Cache into adaptive mode.  When this bit is set to a '0', adaptive mode, the Page Directory Base Register contained in CR3 must be identical across all logical processors.  <b>Note:</b> If the Context ID feature flag, ECX[10], is not set to a '1' after executing the CPUID instruction with EAX = 1, then this feature is not supported and BIOS must not alter the contents of this bit location.

In the CPUID instruction function 1 feature information, bit 10 of ECX register (ECX[10]) has been assigned flag to identify "Context ID feature". The status has changed from Reserved to the following:

ECX [Bits]	Descriptions of Feature Flag Value
10	<b>Context ID.</b> A value of 1 indicates the L1 data cache mode can be set to either adaptive mode or shared mode. A value of 0 indicates this feature is not supported.  See definition of the IA32_MISC_ENABLE MSR Bit 24 (L1 Data Cache Context Mode) for more details.

## V2. BR0# Maximum Hold Time Specification Change

The BR0# maximum hold time has changed to 2 BCLKs. This change will be shown in the “System Bus AC Specifications (Reset Conditions)” table and “System Bus Reset and Configuration Timings” figure of the *Mobile Intel® Celeron® Processor on .13 Micron Process in Micro-FCPGA Package Datasheet*. This change will be incorporated in the next revision of the datasheet.

Currently it states:

**Table 21. System Bus AC Specifications (Reset Conditions)**

T# Parameter	Min	Max	Unit	Figure	Notes
T45: Reset Configuration Signals (A[31:3]#, BR0# INIT#, SMI#) Setup Time	4		BCLKs	13	1
T46: Reset Configuration Signals (A[31:3]#, BR0# INIT#, SMI#) Hold Time	2	20	BCLKs	13	2

**NOTES:**

1. Before the deassertion of RESET#.
2. After clock that deasserts RESET#.

It should state:

**Table 21. System Bus AC Specifications (Reset Conditions)**

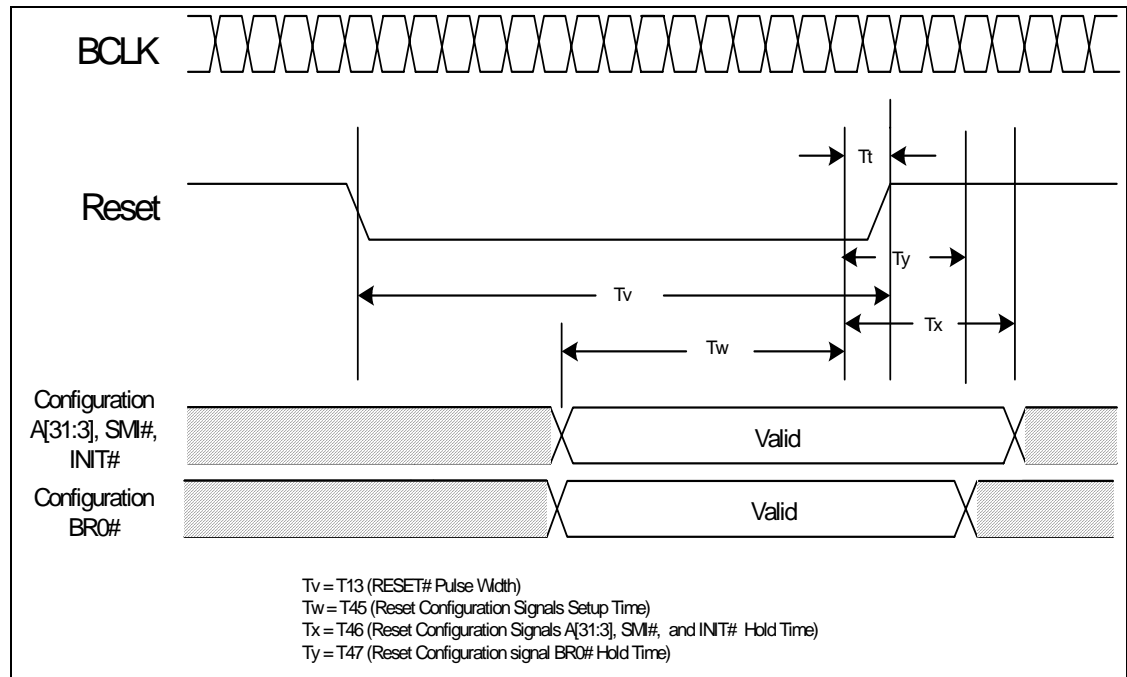
T# Parameter	Min	Max	Unit	Figure	Notes
T45: Reset Configuration Signals (A[31:3]#, BR0# INIT#, SMI#) Setup Time	4		BCLKs	13	1
T46: Reset Configuration Signals (A[31:3]#, INIT#, SMI#) Hold Time	2	20	BCLKs	13	2
T47: Reset Configuration Signal BR0# Hold Time	2	2	BCLKs	13	2

**NOTES:**

1. Before the deassertion of RESET#.
2. After clock that deasserts RESET#.

The following figure will be modified to reflect this change:

**Figure 12. System Bus Reset and Configuration Timings**



§







## Documentation Changes

---

The Documentation Changes listed in this section apply to the following documents:

- *Mobile Intel® Celeron® Processor on .13 Micron Process in Micro-FCPGA Package Datasheet* (Order Number 251308)

**Note:** Documentation changes for IA-32 Intel(R) Architecture Software Developer's Manual volumes 1, 2, and 3 are posted in a separate document *IA-32 Intel(R) Architecture Software Developer's Manual Documentation Changes*. This document has been posted to <http://developer.intel.com/>

All Documentation Changes will be incorporated into a future version of the appropriate mobile Intel Celeron processor on 0.13 micron process in Micro-FCPGA package documentation.

There are no Documentation Changes.

§

